
Analisis Numerik Lanjut - Genetic Algorithm

Ridlo W. Wibowo || 20912009

December 18, 2012

Soal.
Buatlah algoritma genetik untuk menyelesaikan masalah optimasi. Terapkan pada kasus berikut:

- Cari minimum fungsi

$$x^2 - 10\cos(2\pi x) + 10 \qquad -5 \leq x \leq 5$$

- Cari maksimum fungsi

$$x\sin(10\pi x) + 1 \qquad -1 \leq x \leq 2$$

- Cari minimum fungsi

$$f(x_i) = \sum_{i=1}^{2}(x_i - 10\cos(2\pi x_i) + 10) \qquad -5 \leq x_i \leq 5$$

- Cari minimum fungsi

$$f(x_1, x_2) = \frac{1}{2}(x_1^4 - 16x_1^2 + 5x_1) + \frac{1}{2}(x_2^4 - 16x_1^2 + 5x_2) \qquad -4 \leq x_1, x_2 \leq 4$$

Algoritma.
Algoritma Genetika ini kami buat dengan menggunakan urutan proses sebagai berikut:

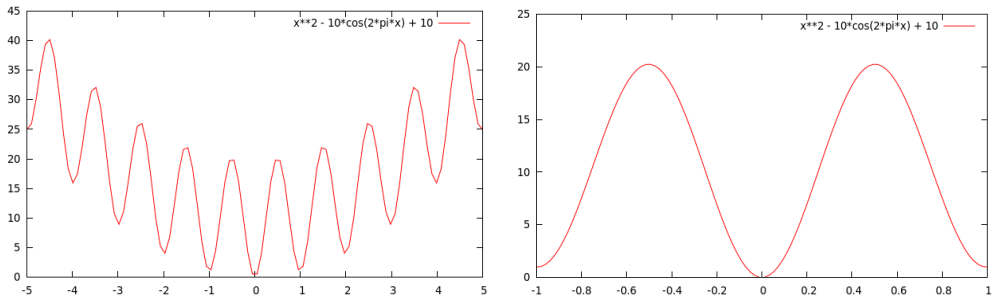
1. INPUT parameter, yaitu sebagai berikut:
   - fungsi $f(x_i)$ yang akan di cari maksimum atau minimumnya
   - batas bawah dan batas atas variabel
   - k_i - ketelitian tiap variabel yang diinginkan (berapa angka dibelakang koma)
   - n - banyaknya khromosom dalam populasi
   - p_c - peluang terjadi persilangan
   - p_m - peluang terjadi mutasi
   - N - banyaknya iterasi
   - *tipe* - 0 untuk pencarian maksimum, 1 untuk pencarian minimum
2. GENERATE vektor biner, sesuai input k (dihitung terlebih dahulu nilai l setiap variabel lalu dijumlahkan), lalu simpan fitness terbaik dari populasi (inisiasi). Menghitung panjang vektor biner untuk k adalah:
   $$l = \frac{\ln((b-a)10^k + 1)}{\ln 2}$$
   dengan a dan b adalah batas bawah dan atas variabel yang dicari.
3. Lakukan proses dibawah ini hingga N kali:
   a) SELECTION, roulette selection dengan menggunakan nilai $f(x)$ yang dinormalisasi (digeser agar positif semua). Untuk mengubah vektor biner menjadi nilai bilangan real dengan cara:
   $$x = a + (\sum_{k=0}^{l-1} b_k 2^k)(\frac{b-a}{2^l - 1})$$
   b) CROSSOVER, dilakukan dengan terlebih dahulu melakukan random untuk memperoleh calon orangtua yang akan disilangkan (menggunakan p_c).
   c) MUTATION, dilakukan terhadap setiap bit dengan memperhatikan p_m.
   d) SAVE, cari nilai fitness terbaik di populasi, lalu bandingkan dengan nilai fitness terbaik yang sudah disimpan, jika lebih baik maka ganti.

   Langkah *Selection* dan *Save* memperhatikan *tipe* masalah.
4. OUTPUT, nilai variabel dan nilai fungsi terbaik (maximum/minimum).

Hasil.
Parameter dibuat sama, yakni: $n = 50, N = 300, p_c = 0.8, p_m = 0.1, k = 5$

- minimum fungsi:

$$x^2 - 10\cos(2\pi x) + 10 \qquad -5 \leq x \leq 5$$



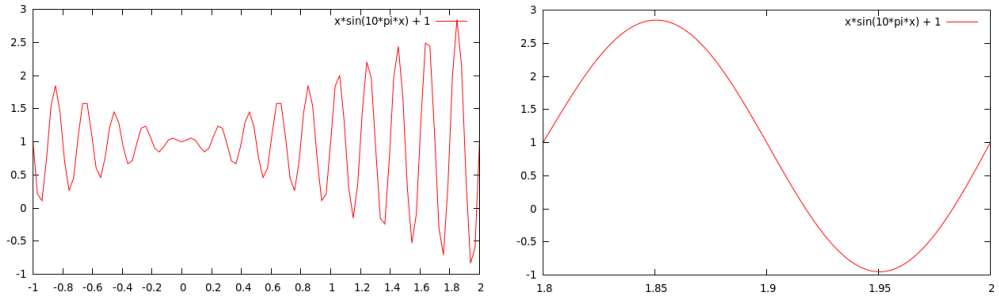
Plot fungsi dan zoom disekitar $x = 0$

running	x_{best}	y_{best}
run_1	4.76838e-06	4.51092e-09
run_2	1.43051e-05	4.05983e-08
run_3	4.76838e-06	4.51092e-09
run_4	-4.76838e-06	4.51092e-09
run_5	2.38419e-05	1.12773e-07

nilai minimum hasil run program berada di sekitar $x = 0$ dan $y = 0$.

- maksimum fungsi:

$$x\sin(10\pi x) + 1 \qquad -1 \leq x \leq 2$$

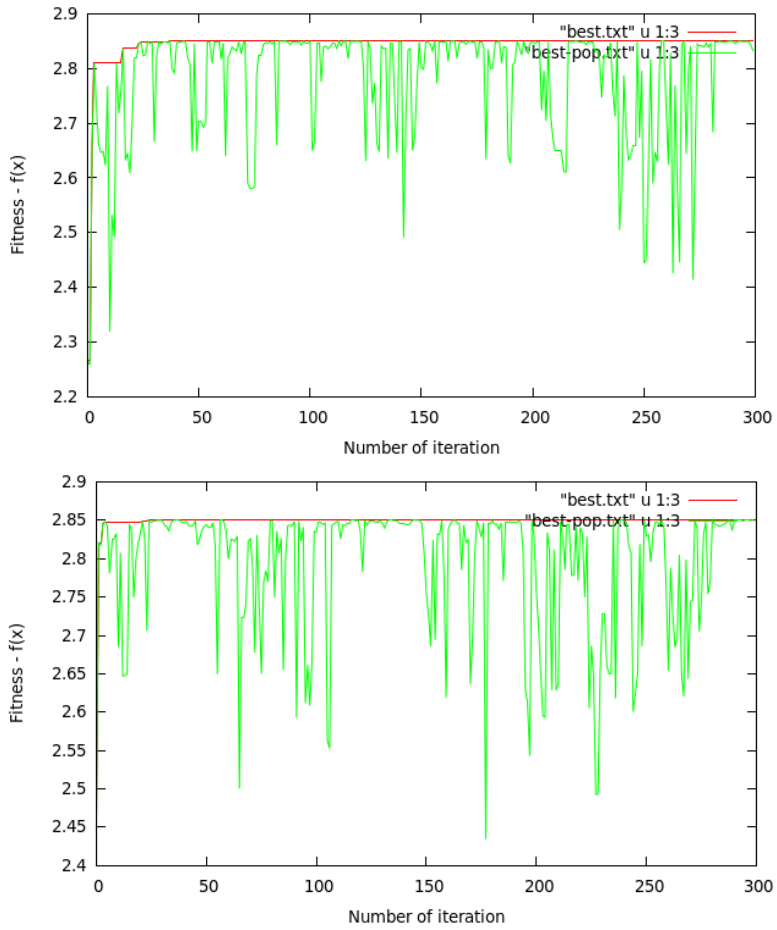


Plot fungsi dan zoom disekitar $x = 1.9$

running	x_{best}	y_{best}
run_1	1.85062	2.85027
run_2	1.85054	2.85027
run_3	1.85056	2.85027
run_4	1.85052	2.85027
run_5	1.85035	2.85024

nilai maksimum hasil run program berada di sekitar $x = 1.85056$ dan $y = 2.85027$.

Contoh screenshot fitness:

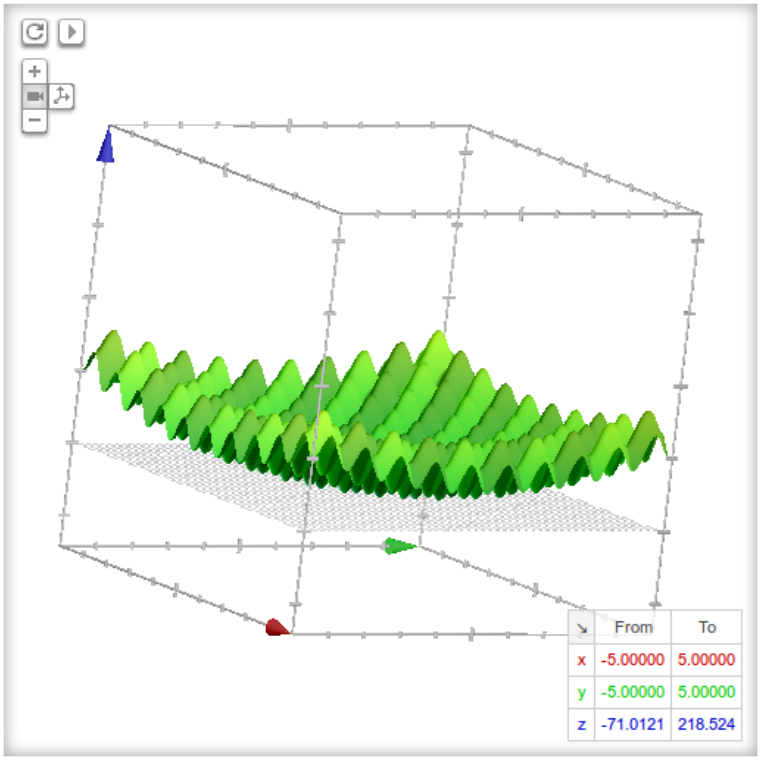


Plot nilai fitness terhadap iterasi, merah menunjukkan fitness terbaik selama iterasi, dan hijau menunjukkan fitness terbaik untuk tiap step iterasi.

untuk kasus 2 dimensi digunakan $N = 500$.

- minimum fungsi:

$$f(x_i) = \sum_{i=1}^{2}(x_i - 10\cos(2\pi x_i) + 10) \qquad -5 \leq x_i \leq 5$$



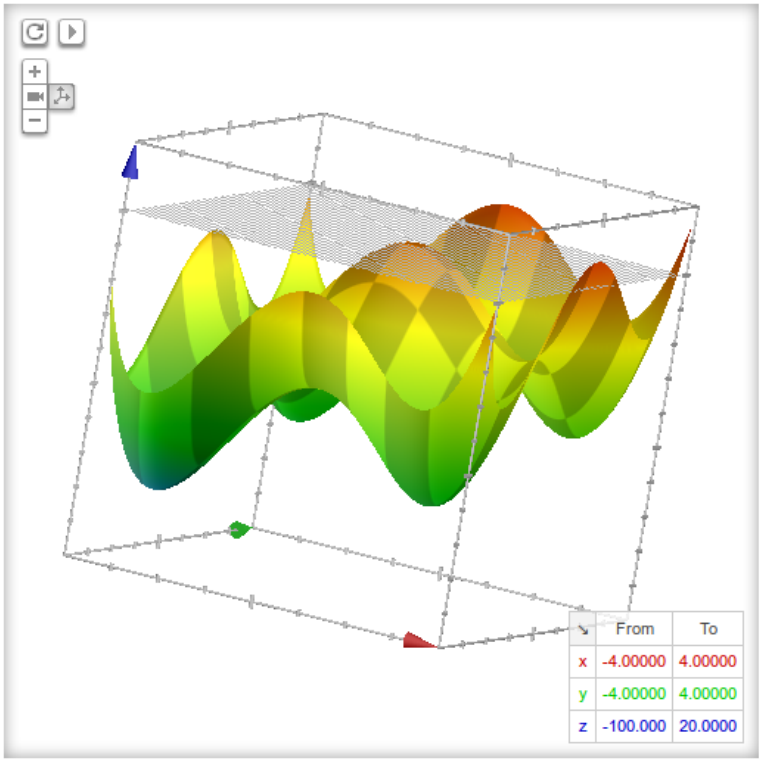
Plot fungsi, minima diduga ditengah kurva (0,0).

running	$x_{1,best}$	$x_{2,best}$	y_{best}
run_1	-0.00058651	1.43051e-05	6.82863e-05
run_2	0.000195503	-0.00110149	0.000248289
run_3	-0.000519753	-0.000414849	8.77374e-05
run_4	0.000681878	-0.000300408	0.000110148
run_5	-0.000414849	-0.000262261	4.77887e-05

nilai minimum hasil run program berada di sekitar $x_1 = 0, x_2 = 0$, dan $f(x_1, x_2) = 0$.

- minimum fungsi:

$$f(x_1, x_2) = \frac{1}{2}(x_1^4 - 16x_1^2 + 5x_1) + \frac{1}{2}(x_2^4 - 16x_1^2 + 5x_2) \quad -4 \leq x_1, x_2 \leq 4$$



Plot fungsi, minima berada di daerah x-negatif dan y-negatif.

running	$x_{1,best}$	$x_{2,best}$	y_{best}
run_1	-2.90181	-2.90195	-78.3322
run_2	-2.9042	-2.90497	-78.3323
run_3	-2.90234	-2.9041	-78.3323
run_4	-2.90497	-2.90304	-78.3323
run_5	-2.90357	-2.90213	-78.3323

nilai minimum hasil run program berada di sekitar $x_1 = -2.903, x_2 = -2.903$, dan $f(x_1, x_2) = -78.3323$.

Lampiran Program.
Kami sampaikan program untuk fungsi n-peubah saja.
Genetic Algorithm - untuk fungsi n-peubah:

```
/*******************************************************/
/* Simple Genetic Algorithm | N-dimensional Function    */
/* Copyleft (c) 2012. Ridlo W. Wibowo                   */
/*******************************************************/

#include <iostream>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#define _USE_MATH_DEFINES
using namespace std;

/***************** FUNCTION ********************/
/* mencari panjang vektor biner minimum */
double minL(double xi, double xf, int ko){
    return log((xf-xi) * pow(10., ko) + 1.)/log(2.);
}

/* random biner */
int brand(){ return rand()%2;}

/* random uniform 0-1 */
double unirand(){ return (double)rand()/(double)RAND_MAX;}

/* menghitung nilai real desimal */
double toDec(double xi, double xf, double li, int khrom[]){
    unsigned int sum = 0;
    for (int i=0;i<li;i++){ sum += khrom[i]*pow(2,i);}
    return (xi + sum*((xf-xi)/(pow(2,li) - 1)));
}

/****************** INPUT PARAMETER ********************/
/* jumlah variabel dalam fungsi */
int v = 2;

/* batas bawah dan atas fungsi */
int u=2*v;
//double bou['u'] = {-5., 5., -5., 5.}; // dua-dua
double bou['u'] = {-4., 4., -4., 4.};

/* ketelitian variable yang dicari (angka di belakang koma) */
int k['v'] = {5, 5};

/* fungsi */
double func(double x[]){
    return (0.5*(pow(x[0],4) - 16*pow(x[0],2) + 5*x[0]) + 0.5*(pow(x[1],4)
        - 16*pow(x[1],2) + 5*x[1]));
    //return (x[0]*x[0] - 10*cos(2.*M_PI*x[0]) + 10. + x[1]*x[1] - 10*cos
        (2.*M_PI*x[1]) + 10.);
}
```

```
49
50
51 /* jumlah khromosom dalam populasi */
52 int n = 50;
53
54 /* peluang terjadi cross-over */
55 double pc = 0.8;
56
57 /* peluang terjadi mutasi */
58 double pm = 0.1;
59
60 /* banyaknya iterasi */
61 int N = 500;
62
63 /* tipe optimasi, 0 = maximisasi, 1 = minimisasi */
64 int tipe = 1;
65
66
67
68 /************* GENETIC FUNCTION ************/
69 /* panjang vektor biner per variabel */
70 double lvar['v'];
71 /* panjang vektor biner total */
72 double l=0;
73 void vecLength(){
74     for (int i=0;i<v;i++){
75         lvar[i] = ceil(minL(bou[(2*i)],bou[(2*i)+1],k[i]));
76         l += lvar[i];
77     }
78 }
79
80 /* populasi */
81 int pop['n']['l'];
82 int induk['n']['l'];
83 int anak['n']['l'];
84
85 /* desimal, variabel dan fitness */
86 double x['n']['v']; double y['n'];
87 double xbest['v']; double ybest;
88 double xbestpop['v']; double ybestpop;
89
90 /* Generate Populasi Awal - biner */
91 void generate(){
92     for (int i=0;i<n;i++){
93         for (int j=0;j<l;j++){
94             pop[i][j] = brand();
95         }
96     }
97 }
98
99 /* mencari nilai fitness */
100 void fitness(){
101     int xpart['n']['v']['l'];
102     for (int i=0;i<n;i++){
```

```
103         int sum1 = 0;
104         int sum2 = 0;
105         for (int j=0;j<v;j++){
106             sum2 = sum1+lvar[j];
107             int o = 0;
108             for (int w=sum1;w<sum2;w++){
109                 xpart[i][j][o] = pop[i][w];
110                 o += 1;
111             }
112             sum1 = sum2;
113         }
114     }
115 
116     for (int i=0;i<n;i++){
117         for (int j=0;j<v;j++){
118             x[i][j] = toDec(bou[(2*j)],bou[(2*j)+1],lvar[j],xpart[i][j]);
119         }
120         y[i] = func(x[i]);
121     }
122 }
123 
124 /* selection, Roulette-Wheel */
125 void selectRoulette(){
126     fitness();
127     double totFit = 0.;
128     double cumFit['n'];
129     double tot = 0.;
130     double fit['n'];
131     double rs;
132 
133     double mini = y[0];
134     for (int i=1;i<n;i++){
135         if (y[i] < mini){ mini = y[i];}
136     }
137 
138     double geser = fabs(mini) + 1.;
139     if (tipe == 0){
140         for (int i=0;i<n;i++){
141             fit[i] = (y[i] + geser);
142          }
143     } // maksimisasi
144     else{for (int i=0;i<n;i++){ fit[i] = 1./(y[i]+geser);}} // minimisasi
145 
146     for (int i=0;i<n;i++){ totFit = totFit + fit[i];}
147     for (int i=0;i<n;i++){
148         cumFit[i] = tot + (fit[i]/totFit);
149         tot = cumFit[i];
150     }
151 
152     for (int i=0;i<n;i++){
153         rs = unirand();
154         for (int j=0;j<n;j++){
155             if (rs <= cumFit[j]){
156                 for (int w=0;w<l;w++){ induk[i][w] = pop[j][w];}
```

```
157                 break;
158             }
159         }
160     }
161 }
162
163 /* cross-over */
164 void crossover(){
165     int parent['n']['l'];
166     int j=0;
167     int p=0;
168     for (int i=0;i<n;i++){
169         if (unirand() <= pc){
170             for (int w=0;w<l;w++){ parent[j][w] = induk[i][w];}
171             j += 1;
172         }
173         else{
174             for (int w=0;w<l;w++){ anak[p][w] = induk[i][w];}
175             p += 1;
176         }
177     }
178     if (j%2 == 1){ // kalau ganjil
179         for (int w=0;w<l;w++){ anak[p][w] = parent[j-1][w];}
180         p += 1;
181         j -= 1;
182     }
183     for (int i=0;i<j/2;i++){
184         int rk = 1 + rand()%((int)l-1);
185         for (int q=0;q<rk;q++){
186             anak[p+(2*i)][q] = parent[(2*i)][q];
187             anak[p+(2*i+1)][q] = parent[(2*i+1)][q];
188         }
189         for (int r=rk;r<l;r++){
190             anak[p+(2*i)][r] = parent[(2*i+1)][r];
191             anak[p+(2*i+1)][r] = parent[(2*i)][r];
192         }
193     }
194 }
195
196 /* mutation */
197 void mutasi(){
198     for (int i=0;i<n;i++){
199         for (int j=0;j<l;j++){
200             if (unirand() <= pm){
201                 if (anak[i][j] == 0){ anak[i][j] = 1;}
202                 else{ anak[i][j] = 0;}
203             }
204         }
205     }
206 }
207
208 /* tukar populasi lama dengan yang baru */
209 void swapper(){
210     for (int i=0;i<n;i++){
```

```
211         for (int j=0;j<l;j++){
212             pop[i][j] = anak[i][j];
213         }
214     }
215 }
216 
217 /* simpan nilai terbaik di populasi */
218 void keep_the_best(){
219     fitness();
220     for (int i=0;i<v;i++){ xbestpop[i] = x[0][i];}
221     ybestpop = y[0];
222     if (tipe == 0){
223         for (int i=1;i<n;i++){
224             if (y[i] > ybestpop){
225                 for (int j=0;j<v;j++){ xbestpop[j] = x[i][j];}
226                 ybestpop = y[i];
227             }
228         }
229     } else{
230         for (int i=1;i<n;i++){
231             if (y[i] < ybestpop){
232                 for (int j=0;j<v;j++){ xbestpop[j] = x[i][j];}
233                 ybestpop = y[i];
234             }
235         }
236     }
237 
238     if (tipe == 0){
239         if (ybestpop > ybest){
240             ybest = ybestpop;
241             for (int j=0;j<v;j++){ xbest[j] = xbestpop[j];}
242         }
243     } else{
244         if (ybestpop < ybest){
245             ybest = ybestpop;
246             for (int j=0;j<v;j++){ xbest[j] = xbestpop[j];}
247         }
248     }
249 }
250 
251 /* print best of the best */
252 void init(){
253     keep_the_best();
254     for (int j=0;j<v;j++){ xbest[j] = xbestpop[j];}
255     ybest = ybestpop;
256 }
257 void print_best(){
258     for (int j=0;j<v;j++){ cout << xbest[j] << " ";}
259     cout << " " << ybest << endl;
260 }
261 void print_best_pop(){
262     for (int j=0;j<v;j++){ cout << xbestpop[j] << " ";}
263     cout << " " << ybestpop << endl;
264 }
```

```
265
266 /* print real variable */
267 void print_real(){
268     for (int i=0;i<n;i++){
269         for (int j=0;j<v;j++){ cout << x[i][j] << " ";}
270         cout << y[i] << endl;
271     }
272     cout << "\n\n";
273 }
274
275
276 /*************** FUNGSI-FUNGSI TESTING *****************/
277 /* print populasi */
278 void print_pop(){
279     for (int i=0;i<n;i++){
280         for (int j=0;j<l;j++){
281             cout << pop[i][j] << " ";
282         }
283         cout << endl;
284     }
285     cout << endl;
286 }
287
288 /* print induk */
289 void print_induk(){
290     for (int i=0;i<n;i++){
291         for (int j=0;j<l;j++){
292             cout << induk[i][j] << " ";
293         }
294         cout << endl;
295     }
296     cout << endl;
297 }
298
299 /* print anak */
300 void print_anak(){
301     for (int i=0;i<n;i++){
302         for (int j=0;j<l;j++){
303             cout << anak[i][j] << " ";
304         }
305         cout << endl;
306     }
307     cout << endl;
308 }
309
310
311 /*************** MAIN ***************/
312 int main(){
313     vecLength();
314     srand(time(NULL));
315     generate();
316     init(); //print_pop();
317     for (int s=0;s<N;s++){
318         selectRoulette();
```

```
319             //print_induk();
320             crossover();
321             //print_anak();
322             mutasi();
323             //print_anak();
324             swapper();
325             keep_the_best();
326             //print_real();
327             //print_best_pop();
328         }
329         cout << "\nBest value: \n";
330         print_best();
331 
332         return 0;
333 }
```